View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Caltech Authors

Angewandte Chemie

Eine Zeitschrift der Gesellschaft Deutscher Chemiker

Supporting Information

# Effect of Two-Dimensional Crystal Orbitals on Fermi Surfaces and Electron Transport in Three-Dimensional Perovskite Oxides

*Maxwell Thomas Dylla, Stephen Dongmin Kang, and G. Jeffrey Snyder**

anie_201812230_sm_miscellaneous_information.pdf

# Supporting Information

## I Anisotropic limit of ellipsoidal pockets: cylinders

Consider a prolate, ellipsoidal carrier pocket. The dispersion of this pocket is characterized by a light mass ($m_l^*$) in two directions and a heavy mass in the third direction ($m_h^*$).

$$\varepsilon = \hbar^2 \left( \frac{k_x^2 + k_y^2}{2m_l^*} + \frac{k_z^2}{2m_h^*} \right) \tag{1}$$

The band edge of this carrier pocket is the point ($k_x = k_y = k_z = 0$) in k-space, and Fermi surfaces of the carrier pocket are prolate ellipsoids. In the limit of an extremely heavy mass ($m_h^* \to \infty$) there is no dispersion in the heavy direction.

$$\varepsilon = \frac{\hbar^2 \left( k_x^2 + k_y^2 \right)}{2m^*} = \frac{\hbar^2 r^2}{2m^*} \tag{2}$$

The band edge of this carrier pocket is now the *line* ($k_x = k_y = 0$), and Fermi surfaces of this carrier pocket are *cylinders*. The length of the cylinder is restricted by the size of the first Brillouin zone. Since the electron group velocity in zero in the heavy mass direction, the conductivity tensor has only two non-zero elements ($\sigma^*$) they are in the directions perpendicular to the cylindrical axis.

$$\sigma_{ij} = \begin{bmatrix} \sigma^* & 0 & 0 \\ 0 & \sigma^* & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3}$$

The transport tensor $\nu$ (which is the numerator of the Seebeck coefficient for isotropic systems) is also anisotropic.

$$\nu_{ij} = \begin{bmatrix} \nu^* & 0 & 0 \\ 0 & \nu^* & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4}$$

We will see that three orthogonal cylindrical pockets give isotropic transport coefficients.

## II Three orthogonal cylindrical pockets

The total conductivity tensor from multiple carrier pockets is given by element-wise addition [1]. The conductivity from three pockets is now isotropic.

$$\sigma_{ij} = \sum_k \sigma_{ij,k} = 2\sigma^* \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Likewise, $\nu$ adds element-wise for multiple carrier pockets.

$$\nu_{ij} = \sum_k \nu_{ij,k} = 2\nu^* \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

The total seebeck coefficient for multiple carrier pockets is then given in terms of $\sigma$ and $\nu$.

$$\alpha_{ij} = \sigma_{\alpha i}^{-1} \nu_{\alpha j} = \frac{\nu^*}{\sigma^*} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \alpha^* \tag{7}$$

Since both the conductivity and Seebeck coefficient are isotropic, we need to only compute one of the tensor elements from a single cylindrical pocket ($\sigma^*$ and $\alpha^*$) to determine the overall transport from three cylindrical pockets.

## III The conductivity tensor element

The conductivity tensor is computed by integrals over the Brillouin zone [1].

$$\sigma_{ij} = e^2 \int \frac{d\mathbf{k}}{4\pi^3} \tau(\varepsilon)\, \mathbf{v}_i(\mathbf{k}) \mathbf{v}_j(\mathbf{k}) \left( -\frac{\partial f}{\partial \varepsilon} \right)_{\varepsilon = \varepsilon(\mathbf{k})}, \tag{8}$$

where $\mathbf{v}_i(\mathbf{k}) = \frac{1}{\hbar} \frac{\partial \varepsilon(\mathbf{k})}{\partial k_i}$ is the group velocity, $\tau$ is the relaxation time, and $f$ is the Fermi-Dirac distribution function. We will first evaluate the conductivity tensor element $\sigma^*$ for a cylindrical pocket.

$$\sigma^* = \sigma_{xx} = e^2 \int \frac{d\mathbf{k}}{4\pi^3} \tau(\varepsilon)\, \mathbf{v}_x(\mathbf{k}) \mathbf{v}_x(\mathbf{k}) \left( -\frac{\partial f}{\partial \varepsilon} \right)_{\varepsilon = \varepsilon(\mathbf{k})}, \tag{9}$$

where

$$\mathbf{v}_x(\mathbf{k}) = \frac{1}{\hbar}\frac{\partial\varepsilon\left(\mathbf{k}\right)}{\partial k_x} \tag{10}$$

$$= \frac{1}{\hbar}\frac{\partial}{\partial k_x}\left[\frac{\hbar^2\left(k_x^2+k_y^2\right)}{2m^*}\right] \tag{11}$$

$$= \frac{1}{\hbar}\frac{2\hbar^2 k_x}{2m^*} \tag{12}$$

$$= \frac{\hbar k_x}{m^*}. \tag{13}$$

$$\implies \sigma^* = e^2\int\frac{d\mathbf{k}}{4\pi^3}\tau\left(\varepsilon\right)\left(\frac{\hbar k_x}{m^*}\right)^2\left(-\frac{\partial f}{\partial\varepsilon}\right)_{\varepsilon=\varepsilon(\mathbf{k})} \tag{14}$$

The integration over all of k-space is easiest in cylindrical coordinates with the z-axis orientated along the cylindrical axis.

$$\implies \sigma^* = e^2\int\int\int\frac{rdrd\theta dz}{4\pi^3}\tau\left(\varepsilon\right)\left(\frac{\hbar r\cos\theta}{m^*}\right)^2\left(-\frac{\partial f}{\partial\varepsilon}\right)_{\varepsilon=\varepsilon(r)} \tag{15}$$

$$= \frac{e^2\hbar^2}{4\pi^3 m^{*2}}\int_{r=0}^{\infty}dr\;\tau\left(\varepsilon\right)r^3\left(-\frac{\partial f}{\partial\varepsilon}\right)_{\varepsilon=\varepsilon(r)}\int_{\theta=0}^{2\pi}d\theta\cos^2\theta\int_{z=0}^{l}dz \tag{16}$$

$$= \frac{e^2\hbar^2 l}{4\pi^2 m^{*2}}\int_{r=0}^{\infty}dr\;\tau\left(\varepsilon\right)r^3\left(-\frac{\partial f}{\partial\varepsilon}\right)_{\varepsilon=\varepsilon(r)} \tag{17}$$

The energy $\varepsilon$ is directly related to $r$ through the $\varepsilon - k$ relationship (Eq. 2).

$$r = \sqrt{\frac{2m^*}{\hbar^2}}\varepsilon^{1/2} \implies dr = \frac{1}{2}\sqrt{\frac{2m^*}{\hbar^2}}\varepsilon^{-1/2}d\varepsilon \tag{18}$$

$$\implies \sigma^* = \frac{e^2\hbar^2 l}{4\pi^2 m^{*2}}\int_{\varepsilon=0}^{\infty}\left(\frac{1}{2}\sqrt{\frac{2m^*}{\hbar^2}}\varepsilon^{-1/2}d\varepsilon\right)\;\tau\left(\varepsilon\right)\left(\sqrt{\frac{2m^*}{\hbar^2}}\varepsilon^{1/2}\right)^3\frac{-\partial f}{\partial\varepsilon} \tag{19}$$

$$= \frac{e^2\hbar^2 l}{4\pi^2 m^{*2}}\int_0^{\infty}\frac{1}{2}\left(\sqrt{\frac{2m^*}{\hbar^2}}\right)^4\left(\varepsilon^{1/2}\right)^2\tau\left(\varepsilon\right)\frac{-\partial f}{\partial\varepsilon}d\varepsilon \tag{20}$$

$$\boxed{\sigma^* = \frac{e^2 l}{2\pi^2\hbar^2}\int_0^{\infty}\varepsilon\;\tau\left(\varepsilon\right)\frac{-\partial f}{\partial\varepsilon}d\varepsilon} \tag{21}$$

Provided an expression for the relaxation time $\tau$ is given, we now have an expression for the conductivity tensor element $\sigma^*$.

## IV The Seebeck tensor element

By calculating the conductivity tensor element $\sigma^*$, we are now able to infer the Seebeck tensor element $\alpha^*$. The Seebeck coefficient is the ratio of two transport integrals. The numerator is the conductivity convoluted with $(\varepsilon - \mu)$, which is $\nu^*$. The denominator is expressly conductivity. There is an addition factor of $1/eT$ that comes from the numerator [1].

$$\boxed{\alpha^* = \frac{1}{eT} \frac{\int_0^\infty \varepsilon(\varepsilon - \mu)\, \tau(\varepsilon) \dfrac{-\partial f}{\partial \varepsilon} d\varepsilon}{\int_0^\infty \varepsilon\, \tau(\varepsilon) \dfrac{-\partial f}{\partial \varepsilon} d\varepsilon}} \tag{22}$$

We now have equations for commonly measured transport coefficient (conductivity and Seebeck) from a cylindrical carrier pocket.

## V Density of states and carrier concentration

The number of states $N$ in a cylindrical volume of k-space is

$$N = 2\frac{\pi l r^2}{(2\pi)^3}, \tag{23}$$

where $r$ is the radius of the cylinder and $l$ is its length. We have expressly included the electron spin degeneracy. With the $\varepsilon - k$ relationship (Eq. 2) we convert radius to energy:

$$\implies N = \frac{2\pi l}{(2\pi)^3} \frac{2m^*\varepsilon}{\hbar^2} \tag{24}$$

$$= \frac{2lm^*}{(2\pi)^2 \hbar^2} \varepsilon. \tag{25}$$

The density of states $\mathcal{N}$ is the number of states $dN$ within an energy window $d\varepsilon$.

$$\boxed{\mathcal{N}(\varepsilon) = \frac{dN}{d\varepsilon} = \frac{lm^*}{2\pi^2\hbar^2}} \tag{26}$$

The density of states for a cylindrical pocket is independent of energy, which is like the density of states for a 2D, quantum-confined system. The convolution of $\mathcal{N}$ with the Fermi-Dirac distribution function determines the number of carriers $n$.

$$n = \int_0^\infty \mathcal{N}(\varepsilon) f(\varepsilon) d\varepsilon \tag{27}$$

$$= \int_0^\infty \frac{lm^*}{2\pi^2\hbar^2} f(\varepsilon) d\varepsilon \tag{28}$$

$$\boxed{n = \frac{lm^*}{2\pi^2\hbar^2} \int_0^\infty f(\varepsilon) d\varepsilon} \tag{29}$$

# VI Carrier relaxation time

The goal of this section is to derive an energy-dependent relaxation time in a deformation potential framework from Fermi's golden rule. Fermi's golden rule determines the transition frequency $W$ from initial state $\mathbf{k}$ to final state $\mathbf{k}'$. Each initial and final state are connected by a matrix element $M$, and energy must be conserved.

$$W_{\mathbf{k},\mathbf{k}'} = \frac{2\pi}{\hbar} \left|M_{\mathbf{k},\mathbf{k}'}\right|^2 \delta(E_\mathbf{k} - E_{\mathbf{k}'}) \tag{30}$$

The matrix element $M$ is represented as a convolution of the initial and final states with a perturbing potential $V$, which is (in general) position ($\mathbf{r}$) dependent. The initial and final states are represented as plane waves $\left(|\mathbf{k}\rangle \leftrightarrow e^{i\mathbf{k}\cdot\mathbf{r}}\right)$.

$$M_{\mathbf{k},\mathbf{k}'} = \langle \mathbf{k}'|H'|\mathbf{k}\rangle = \int V(\mathbf{r}) e^{i(\mathbf{k}-\mathbf{k}')\cdot\mathbf{r}} dV \tag{31}$$

Each final state $\mathbf{k}'$ additively contributes to the total scattering frequency $\Gamma$ for a particular initial state $\mathbf{k}$; the energy-conserving $\delta$-function in Eq. 30 heavily restricts the number of allowed final states $\{\mathbf{k}'\}$. In addition, scattering events are weighed by a forward scattering factor. The scattering frequency is inversely related to the relaxation time $\tau$.

$$\Gamma(\mathbf{k}) = \frac{1}{\tau(\mathbf{k})} = \int \frac{d\mathbf{k}'}{(2\pi)^3} W_{\mathbf{k},\mathbf{k}'} \left(1 - \hat{\mathbf{k}}\cdot\hat{\mathbf{k}}'\right) \tag{32}$$

$$= \frac{2\pi}{\hbar} \int \frac{d\mathbf{k}'}{(2\pi)^3} \left|M_{\mathbf{k},\mathbf{k}'}\right|^2 \left(1 - \hat{\mathbf{k}}\cdot\hat{\mathbf{k}}'\right) \delta(E_\mathbf{k} - E_{\mathbf{k}'}) \tag{33}$$

The allowed final states lie on the same isoenergy surface as the initial state (Fermi surface, $S$). In addition, when scattering is by non-polar phonons, the matrix element squared is proportional to the phonon wavevector, which induces the transition ($|M|^2 \sim q = \left|\mathbf{k}' - \mathbf{k}\right|$) [2].

$$\frac{1}{\tau(\mathbf{k})} \sim \frac{2\pi}{\hbar} \int_S \frac{d\mathbf{k}'}{(2\pi)^3} q \left(1 - \hat{\mathbf{k}}\cdot\hat{\mathbf{k}}'\right) \tag{34}$$

However, at finite temperature, the occupation of phonons is described by the Bose-Einstein distribution function $n_q(q) = 1/\left(e^{E(q)/kT} - 1\right)$. The distribution function weights each individual scattering event (from the initial state $\mathbf{k}$ to the final state $\mathbf{k}'$) by the number of phonons which can cause that transition.

$$\frac{1}{\tau(\mathbf{k})} \sim \frac{2\pi}{\hbar} \int_S \frac{d\mathbf{k}'}{(2\pi)^3} n_q q \left(1 - \hat{\mathbf{k}} \cdot \hat{\mathbf{k}}'\right) \tag{35}$$

With knowledge of the electron Fermi surface (which sets $\{\mathbf{k}'\}$, the set of final states) and the phonon dispersion (which sets $n_q$, the phonon occupation), the relaxation time can be computed numerically.

$$\boxed{\frac{1}{\tau(\mathbf{k})} \sim \sum_{\{\mathbf{k}'\} \in S} n_q q \left(1 - \hat{\mathbf{k}} \cdot \hat{\mathbf{k}}'\right)} \tag{36}$$

Scattering times were computed with Equation 36 for spherical and cylindrical Fermi surfaces. A Debye phonon dispersion ($\hbar\omega = v_s q$) was used to illustrate how the shape of the Fermi surface impacts the temperature dependence of the relaxation time. The results are shown in Figure 1. At very low temperatures, both Fermi surfaces show Block $T^5$ behavior in the relaxation time. However, near 10 K, the two relaxation times diverge from one another. The scattering frequency rises faster in the cylindrical (2D) case, because additional scattering states along the Fermi surface became accessible in this temperature range, while the final states in the spherical (3D) case are all accessible (because the Fermi surface is not elongated). This model predicts that the $T^2$ behavior in the cylindrical case should become $T^1$ at very high temperatures.

The temperature dependencies of relaxation times are directly controlled through the occupation function $n_q$. However, it is not immediately obvious what this temperature dependency is, because the occupation is a complicated function. The description is significantly simplified by decoupling the occupation statistics into two terms, following the discussion of $T^2$ resistivity in Bi [3]. These two terms represent the fact that low energy phonons are increasing their occupation linearly ($n_\mathrm{q}$), while higher energy phonons are only just becoming occupied at higher temperatures ($n_\mathrm{final}$). For a Debye model dispersion, the front of the final states moves linearly with temperature ($q_\mathrm{thermal} = kT/v_s$, where $v_s$ is the speed of sound). Decoupling the occupation statistics into these two terms illustrates how finite temperature influences the relaxation time (see the discussion of temperature-squared resistivity in the main text). These terms make it clear why low-dimensional Fermi surfaces could have additional temperature dependencies in the relaxation time at elevated temperatures (from the $n_\mathrm{final}$).

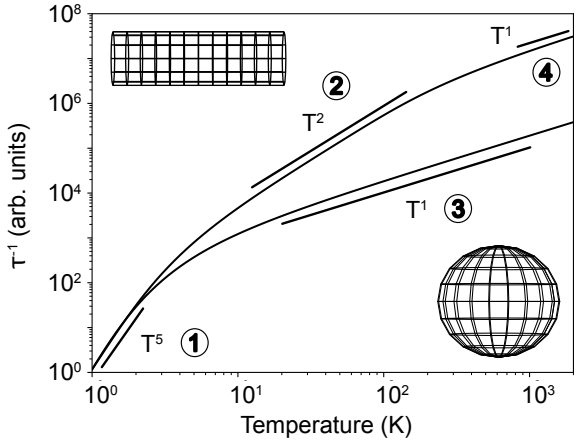


Figure 1: The scattering frequency for electrons on 3D and 2D Fermi surfaces are qualitatively different and explained by the same scattering model. **1)** At low temperatures both types of electrons follow the Block $T^5$ law. **2)** When the phonon Bose sphere exceeds the radius of the 2D Fermi surface but is smaller than Fermi surface length, $T^2$ behavior is observed. **3)** When the phonon Bose sphere exceeds the radius of the 3D Fermi surface, $T^1$ behavior is observed. **4)** The 2D case is also expected to recover $T^1$ behavior when all phonon modes are saturated.

# VII scattering.py

```
'''
the purpose of this module is to compare scattering between cylindrical and
spherical fermi surfaces. we compare the temperature dependence of the
relaxation time between cylindrical and spherical fermi surfaces.
'''

import numpy as np

K_B = 1.3806485e-23 # Boltzmann constant in J/K
H_BAR = 1.05457180e-34 # Planck constant in Js


def bose_einstein(T, v_s, q):
    '''
    Bose-Einstein distribution. Boson energy is parameterized by wavevectors
    in the Debye approximation.

    Args:
        T (float):
            the temperature in K
        v_s (float):
            the speed of sound in m/s
        q (float):
            magnitude of the phonon wavevector in 1/m

    Returns: float
        the phonon occupation
    '''

    return 1. / (np.exp(v_s * q * H_BAR / K_B / T) - 1.)


def forward_scattering(k, k_prime):
    '''
    forward scattering term for an initial state connected to an array of
    final states. (one k connected to many k')

    Args:
        k (ndarray 1D)
            vector of the initial state
        k_prime (ndarray 2D)
            array of vectors of the final states
```

```
    Return:
        forward scattering (1 - dot(k, k')/|k||k'|)
    '''

    k = np.array([k for item in k_prime])
    initial_norms = np.sqrt((k * k).sum(axis=1))
    final_norms = np.sqrt((k_prime * k_prime).sum(axis=1))
    return (1. - np.einsum('ij,ij->i', k, k_prime) / initial_norms /
            final_norms)


class Cylinder(object):
    '''
    representation of a cylindrical Fermi surface
    '''

    def __init__(self, length, radius):
        '''
        Args:
            length (float):
                the length of the cylinder in 1/m
            radius (float):
                the radius of the cylinder in 1/m
        '''
        self.length = length
        self.radius = radius

    def get_surface_catesian(self, divisions):
        '''
        return a matrix of equally spaced points on the surface of the cylinder
        in cartesian coordinates

        Args:
            divisions (int):
                the density from which to sample the surface

        Returns: np.ndarray
            rows - points on cylinder
            columns - [x y z]
        '''

        theta = np.linspace(0, 2 * np.pi, num=divisions, endpoint=False)
        z = np.linspace(-self.length / 2., self.length / 2., num=divisions,
                        endpoint=False)
        cylindrical_vectors = np.stack(np.meshgrid([self.radius], theta, z),
                                       -1).reshape(-1, 3)
```

```
        x = cylindrical_vectors[:, 0] * np.cos(cylindrical_vectors[:, 1])
        y = cylindrical_vectors[:, 0] * np.sin(cylindrical_vectors[:, 1])
        z = cylindrical_vectors[:, 2]

        return np.vstack((x, y, z)).T


class Sphere(object):
    '''
    representation of a spherical Fermi surface
    '''

    def __init__(self, radius):
        '''
        Args:
            radius (float):
                the radius of the sphere in 1/m
        '''
        self.radius = radius

    def get_surface_catesian(self, divisions):
        '''
        return a matrix of equally spaced points Iin angle)on the surface of
        the sphere in cartesian coordinates

        Args:
            divisions (int):
                the density from which to sample the surface

        Returns: np.ndarray
            rows - points on cylinder
            columns - [x y z]
        '''

        # theta is inclination, phi is azimuthal angle
        theta = np.linspace(0, np.pi, num=divisions, endpoint=False)[1:]
        phi = np.linspace(0, 2 * np.pi, num=divisions, endpoint=False)
        spherical_vectors = np.stack(np.meshgrid([self.radius], theta, phi),
                                     -1).reshape(-1, 3)
        x = (spherical_vectors[:, 0] * np.sin(spherical_vectors[:, 1]) *
             np.cos(spherical_vectors[:, 2]))
        y = (spherical_vectors[:, 0] * np.sin(spherical_vectors[:, 1]) *
             np.sin(spherical_vectors[:, 2]))
        z = spherical_vectors[:, 0] * np.cos(spherical_vectors[:, 1])

        return np.vstack((x, y, z)).T
```

## VIII relaxation_time.py

```
'''
We assume elastic scattering where energy and momentum is conserved in electron
phonon processes. We also assume that the matrix element squared (|M|^2) is
proportional to the phonon wavecector (q). Below are estimations of the
Fermi-surface dimentions, which are used as the inputs of this model.

cylindrical length -- the unit cell of SrTiO3 is 3.905 A. since the unit-cell
is cubic and the Fermi-surface cylinders are orientated along Gamma-X
the cylindrical length (l) is computed as:
    l = 2 * (2 * pi) / (2 * 3.905 A) = 1.609e10 1/m

cylindrical radius -- in a three cylinder model, where the curvature mass is
0.5 m_e and the cylindrical length is 1.609e10 1/m; the cylindrical Fermi
wavevector (k_F) is computed to be the following by the carrier concentration
expression derived in the supplemental. We assume n=1e19 cm^-3:
    k_F = sqrt(1e25 / 3. * (2 * np.pi)**2 / 1.609e10) = 0.00904e10 1/m
'''

from scattering import *

import numpy as np
import matplotlib.pyplot as plt

# set-up of parameters
temperatures = np.logspace(0, 4, 100) # K
v_s = 7900 # m/s
fermi_radius = 0.00904e10 # 1/m
bz_length = 1.609e10 # 1/m
surface_density = 1001 # density of points on Fermi surfaces

# relaxation time for cylinder with acoustic branch
print('calculating tau for cylinder (acoustic)')

# obtain initial and final wavevectors for scattering
fermi_surface = Cylinder(bz_length, fermi_radius)
inital_k = np.array([fermi_surface.radius, 0., 0.])
final_k = fermi_surface.get_surface_catesian(surface_density)

# do not count the initial electronic state
try:
    index = final_k.tolist().index(inital_k.tolist())
    final_k = np.delete(final_k, index, 0)
except ValueError:
```

```
    pass

# compute the relaxation time for different temperatures
q_vector_mags = np.sqrt(((final_k - inital_k) *
                      (final_k - inital_k)).sum(axis=1))
numerator = (final_k - inital_k) * (final_k - inital_k)
numerator = numerator[:, 0] + numerator[:, 1] + numerator[:, 2]
forward_scattering_term = 1. - (final_k[:, 0] / inital_k[0])
relaxation_times = []
for T in temperatures:
    occupation_term = bose_einstein(T, v_s, q_vector_mags)
    relaxation_times.append(sum(numerator * occupation_term *
                                forward_scattering_term / q_vector_mags))
cylinder_0_K = relaxation_times[0]
plt.plot(temperatures, 1. / np.array(relaxation_times) * 1e9, 'k-')

# relaxation time for cylinder with acoustic and optical branch
print('calculating tau for cylinder (acoustic + optical)')

# obtain initial and final wavevectors for scattering
fermi_surface = Cylinder(bz_length, fermi_radius)
inital_k = np.array([fermi_surface.radius, 0., 0.])
final_k = fermi_surface.get_surface_catesian(surface_density)

# do not count the initial electronic state
try:
    index = final_k.tolist().index(inital_k.tolist())
    final_k = np.delete(final_k, index, 0)
except ValueError:
    pass

# compute the relaxation time for different temperatures
q_vector_mags = np.sqrt(((final_k - inital_k) *
                      (final_k - inital_k)).sum(axis=1))
numerator = (final_k - inital_k) * (final_k - inital_k)
numerator = numerator[:, 0] + numerator[:, 1] + numerator[:, 2]
forward_scattering_term = 1. - (final_k[:, 0] / inital_k[0])
relaxation_times = []
for T in temperatures:
    acoustic_occupation = bose_einstein(T, v_s, q_vector_mags)
    optical_occupation = bose_einstein(T, v_s, q_vector_mags + (bz_length / 2))
    relaxation_times.append(sum(numerator * acoustic_occupation *
                                forward_scattering_term / q_vector_mags) +
                            10 * sum(numerator * optical_occupation *
                                     forward_scattering_term / q_vector_mags))
cylinder_0_K = relaxation_times[0]
```

```
plt.plot(temperatures, 1. / np.array(relaxation_times) * 1e9, 'r-')


# display comparison between acoustic and acoustic + optical scattering
plt.xscale('log')
plt.yscale('log')
plt.xlabel('T (K)')
plt.ylabel(r'$\tau^{-1}$ (arbitrary unit)')
plt.xlim(100, 1000)
plt.ylim(1e-8, 1e-4)
plt.savefig('tau.pdf', dpi=300, format='pdf')
plt.show()
```

## References


[1] Georg K. H. Madsen and David J. Singh. BoltzTraP. A code for calculating band-structure dependent quantities. *Computer Physics Communications*, 175(1):67–71, 2006.

[2] J. Bardeen and W. Shockley. Deformation Potentials and Mobilities in Non-Polar Crystals. *Physical Review*, 80(1):72–80, October 1950.

[3] Carl A. Kukkonen. $T^2$ electrical resistivity due to electron-phonon scattering on a small cylindrical Fermi surface: Application to bismuth. *Physical Review B*, 18(4):1849–1853, August 1978.